03 – More on Commands

CS 2043: Unix Tools and Scripting, Spring 2019 [1]

Matthew Milano
January 28, 2019

Cornell University
1. Flags & Command Clarification
Flags & Command Clarification
Most commands take flags and optional arguments. These come in two general forms:

- Switches (no argument required), and
- Argument specifiers (for lack of a better name).

When specifying flags for a given command, keep in mind:

- Flags modify the behavior of the command / how it executes.
- Some flags take precedence over others, and some flags you specify can implicitly pass additional flags to the command.

There is no absolute rule here: research the command.
• A flag that is
  • One letter is specified with a single dash (-a).
  • More than one letter is specified with two dashes (--all).
  • The reason is because of how switches can be combined.

• We generally use “flag” and “switch” interchangeably:
  • “flag” the command, telling it that “action X” should occur
  • specify to the command to “switch on/off action X”
Switches take no arguments, and can be specified in a couple of different ways.
Switches are usually one letter, and multiple letter switches usually have a one letter alias.

One option:
- `ls -a`
- `ls --all`

Two options:
- `ls -l -Q`
- `ls -lQ`

Usually applied from left to right in terms of operator precedence, but not always:
- This is up to the developer of the tool.
- Prompts: `rm -fi <file>`
- Does not prompt: `rm -if <file>`
The `--argument="value"` format, where the `=` and quotes are needed if `value` is more than one word.

- Yes: `ls --hide="Desktop" ~/`
- Yes: `ls --hide=Desktop ~/`
  - One word, no quotes necessary
- No: `ls --hide = "Desktop" ~/`
  - Spaces by the `=` will be misinterpreted
  - It used `=` as the argument to `hide`

The `--argument value` format (space after the `argument`).

- Quote rules same as above.
- `ls --hide "Desktop" ~/`
- `ls --hide Desktop ~/`

Usually, `--argument value` and `--argument=value` are interchangeable.

- Not always!
Generally, always specify the flags before the arguments.

- `ls -l ~/Desktop/` and `ls ~/Desktop/ -l` both work.
  - Sometimes flags after arguments get ignored.
  - Depends both on the command, and the flag(s).

- The special sequence `--` signals the end of the options.
  - Executes as expected: `ls -l -a ~/Desktop/`
  - Only uses `-l`: `ls -l -- -a ~/Desktop/`
    - "`ls: cannot access -a: No such file or directory`
    - The `-a` was treated as an argument, and there is no `-a` directory (for me)

- In this example:
  - `-l` and `-a` are the flags.
  - `~/Desktop/` is the argument.
• The special sequence `--` that signals the end of the options is often most useful if you need to do something special.

• Suppose I wanted to make the folder `-a` on my Desktop.

```bash
$ cd ~/Desktop # for demonstration purpose
$ mkdir -a      # fails: invalid option -- 'a'
$ mkdir -- -a   # success! (ls to confirm)
$ rmdir -a      # fails: invalid option -- 'a'
$ rmdir -- -a   # success! (ls to confirm)
```

• This trick can be useful in many scenarios, and generally arises when you need to work with special characters of some sort.
• How do I know what the flags / options for all of these commands are?

The **Manual Command**

```
man command_name
```

- Loads the manual (manpage) for the specified command.
- Unlike google, manpages are **system-specific**.
- Usually very comprehensive. Sometimes *too* comprehensive.
- Type `/keyword` to search for `keyword`, and hit `<enter>`.
- The `n` key jumps to the next search result.

• Search example on next page if that was confusing. Intended for side-by-side follow-along.
Man oh man

- The **man** command is really useful!

  ```
  $ man man  # you now have the manual loaded
  $ /useful  # type /useful, then hit enter
  ####### [[[ first result highlighted ]]]
  $ n  # followed by enter
  ####### [[[ next result highlighted ]]]
  # The default 'pager' is `less`, type `q`
  # without backticks to exit.
  ```

- Subtle differences depending on distribution, e.g. `ls -B`
- BSD/OSX: Force printing of non-printable characters in file names as `\xxx`.
  - `xxx` is the numeric value of the character in **octal**.
- GNU (Fedora, Ubuntu): don’t list implied entries ending with `~`
  - Files ending with `~` are *temporary* backup files that certain programs generate (e.g. some text-editors, your OS).
References

[1] Stephen McDowell, Bruno Abrahao, Hussam Abu-Libdeh, Nicolas Savva, David Slater, and others over the years. “Previous Cornell CS 2043 Course Slides”.